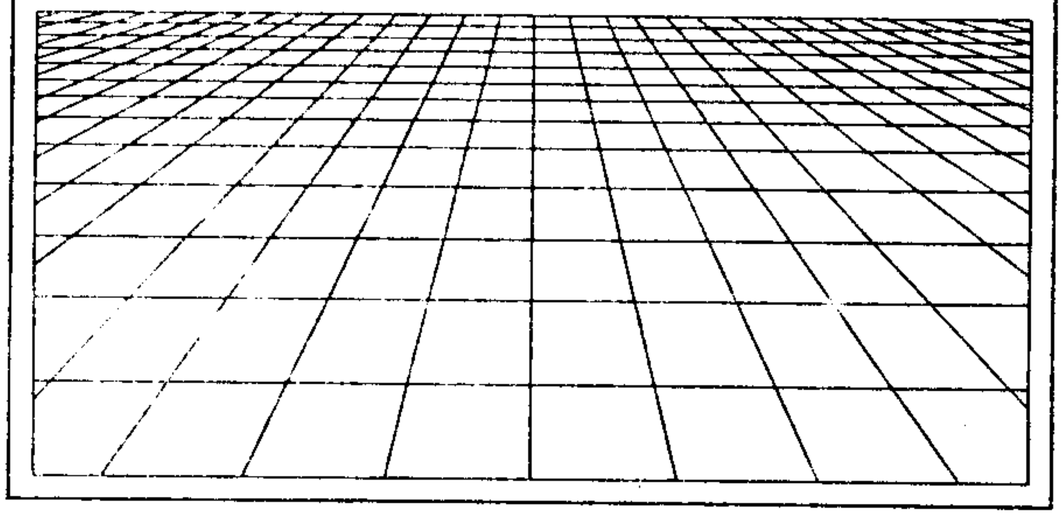
REPUBLIC SOFTWARE™
P.O. Box 23042
L'Enfant Plaza
Washington, D.C. 20024

Utilities I

Four Software Utilities for the Texas Instruments 99/4 or 99/4A Home Computer.



Includes:

DISASSEMBLER—
executes in Extended BASIC *or* in console BASIC if the Mini-Memory or Editor Assembler is available.

POINT-PLOTTING ROUTINES—
high-resolution plotting capability in console BASIC, Extended BASIC, and Assembly Language.

SCREEN DUMP ROUTINES—
written in console BASIC, Extended BASIC, and Assembly Language for the TI 80-column printer and Epson Printers with Graftrax or Graftrax Plus.

SPEECH UTILITY—
allows the speech synthesizer to speak *any* word you wish in your Extended BASIC programs – without TI's Terminal Emulator II or Text-to-Speech programs.

REPUBLIC SOFTWARE

Table of Contents

Congratulations! You have just purchased a high-quality software package that can adapt itself to your computer system's capabilities as your system grows. Each UTILITIES I disk or tape contains programs that will execute on a Texas Instruments 99/4 or 99/4A Home Computer in console BASIC, as well as programs and subprograms that take advantage of the greater speed or capability provided by Extended BASIC, the expansion RAM (Random Access Memory), the speech synthesizer, and the RS232 interface with compatible printer. We believe that the availability of all this programming power for one low price makes UTILITIES I one of the best software values on the market today.

Detailed instructions and equipment requirements for each utility have been included in this manual, and we encourage you to read these directions carefully before attempting to use any of the UTILITIES I progams or subprograms. If you study the instructions thoroughly and perform some of the suggested practice exercises, you will find that this manual can be a helpful guide to "getting the most" out of your software.

Important Note

INTRODUCTION

Each major section of this manual describes a utility, lists required and optional equipment, and gives appropriate directions for using the utility. However, some instructions and procedures are common to all of the utilities and this section has been included to provide that information.

Required Equipment

- Texas Instruments 99/4 or 99/4A Home Computer
- Texas Instruments monitor **OR** RF modulator and television set
- Compatible cassette tape recorder and connecting cables **OR** Texas Instruments Disk Memory System

The lists of required and optional equipment included in each section are intended to describe hardware needed in addition to the items listed above.

Getting Started

Listed below are the steps necessary for turning on a Texas Instruments 99/4 or 99/4A Home Computer and a complete set of optional equipment. Some of the information here may seem trivial, but the order of the steps is important. Please execute these procedures in order (ignoring any references to equipment not used by your own system) before attempting to load any UTILITIES I programs or subprograms.

1. Switch on the cassette tape recorder, and make sure that it is connected to the console as CS1.
2. Switch on the disk drive(s).
3. Switch on the Peripheral Expansion Box and/or any stand-alone peripherals you may have (e.g., Expansion RAM, RS232, or Disk Controller).
4. Switch on the console.
5. Turn on the monitor or television set.
6. Switch on the printer.

7. Insert any necessary modules into the console (e.g., Extended BASIC, Editor/Assembler, or MiniMemory).
8. Insert the UTILITIES I cassette into the tape recorder with the label facing out **OR** insert the UTILITIES I disk into disk drive 1 and close the front cover of the disk drive.
9. When the computer's title screen appears, press any key on the keyboard.
10. Select the appropriate computer language: Extended BASIC (if available) or console BASIC.

Loading Programs and Subprograms

The cassette-based UTILITIES I has the various programs and subprograms recorded in the same order that the programs are described in this manual. Each program is preceded on the tape by a voice recording of its title. No "footage counter" indications are given in this manual because the numbers vary so widely from tape recorder to tape recorder that such recommendations would be useless. However, we suggest that you reset the "footage counter" to "000" on your tape recorder and listen to the tape all the way through once, marking down the counter settings as each program is announced. This will only take a few minutes and will make it much easier to locate specific programs in the future. You will probably need to unplug the "monitor" (white) wire from your recorder in order to listen to the tape. Don't forget to plug this wire back into your recorder before you attempt to load any programs! To load any program or subprogram, locate it on the tape. Listen for the announced title to confirm its location. Then enter OLD CS1 and follow the loading instructions that appear on the screen, with the following exception: Do not rewind the tape when the screen instructions advise you to do so; you are already at the proper tape location.

The disk-based UTILITIES I has the programs and subprograms recorded in alphabetical order. The appropriate title for loading the program is contained in parentheses following the title of the section (e.g., DISASSEMBLER (DIS)). Unless otherwise noted in the instructions for a utility, load any program by entering OLD DSK1.title (e.g., OLD DSK1.DIS).

You are now ready to proceed with the utility of your choice.

DISASSEMBLER (DIS)

Required Equipment

- Extended BASIC **OR** Editor/Assembler **OR** MiniMemory

Optional Equipment

- Expansion RAM
- RS232 Interface and printer

Preliminary Information

Disassemblers are designed to translate the on and off bits (Machine Language) in a computer's memory into the pseudo-English of Assembly Language. This disassembler is intended primarily as a tool for Assembly Language programmers. It should be especially useful to programmers using the cassette-based assembler sold with the MiniMemory, who can use it to print otherwise unobtainable listings of their own Assembly Language programs.

However, if you are not already conversant with Assembly Language, don't despair. This program can be used to help you learn more about the subject. A tutorial concerning Assembly Language is beyond the scope of this manual, but the instructions below contain useful information for beginners, and Texas Instruments now sells several manuals and texts that can also be quite helpful.

The Texas Instruments 99/4 and 99/4A computers and this disassembler are capable of directly "addressing" (see below) 65,536 (64K) 8-bit bytes of memory. In addition to the 64K of directly addressable memory, the 99/4 and 99/4A computers can use the 16K of Video Display Processor (VDP) Random Access Memory (RAM) that is built into the console. The disassembler is not designed to look into the 16K VDP RAM because this area of memory cannot normally be used to store Assembly Language programs.

Each byte of the 64K memory has an "address", or number, that designates that byte and that byte only. The first 32K (32,768) bytes (called "low memory") are numbered 0 through 32767. The second 32K bytes (called "high memory") are

numbered (for reasons having to do with the representation of numbers in computer memory) -32768 through -1. Zero is the bottom byte of the 64K, and -1 is the top byte. Half of this 64K area is the 32K Expansion RAM (8192 to 16383 and -24576 to -32) and the rest is the console Read Only Memory (ROM), cartridge memory, and peripheral device ROM. If the Expansion RAM is present, the high end of it may contain a BASIC program, but most of the rest of the 64K, RAM and ROM alike, is used for Machine Language programs that you can translate into Assembly Language by using this disassembler.

Instructions

Make sure that one of the required cartridges is properly inserted into the console. Select Extended BASIC, if available; otherwise select console BASIC. Load the program normally. Enter RUN.

After the logo screen appears and clears, the program asks for your printer characteristics. If you have no printer, just press the "enter" key to proceed. If you do have a printer, enter its proper title (e.g., RS232 2 or RS232.BA=4800.PA=N.DA=8). Quotation marks are unnecessary. If your printer's description is lengthy, don't stop typing when you reach the end of the input line—just keep typing and your input will wrap around to the next line. Don't press the "enter" key until you have finished the description.

Next, the program asks for the starting address. This is just the memory address of the byte at which you wish to start disassembling. Since the 99/4 and 99/4A are 16-bit computers, the computer and your disassembler expect to look at two bytes at a time starting with an **even** numbered address. The computer has firmware to keep it on track, but the disassembler gets its direction from you. Please enter an **even** number for a starting address. Negative 2 and 0 are even numbers.

The program will now ask you for an ending address. Please enter an address that is numerically greater than the starting address. An odd number will be accepted by the program as if you had entered the next lower even number. Negative 2 is greater than -4, and any positive number is greater than any negative number.

As soon as you have entered the ending address, the disassembling will proceed. Output will always be directed to the

screen. If you have described a printer to the program, output will also be directed to the printer. If you wish to temporarily stop disassembly at any point, press and hold the space bar until printing halts. When you are ready to continue, press the "enter" key. If you wish to stop during disassembly and enter new starting and ending addresses, press the "E" key.

The output of the disassembler, whether on the screen or a printer, consists of four columns. The first column contains the decimal addresses of the area of memory you are disassembling, starting and ending at the addresses you have entered. The second column contains the hexadecimal representation of the addresses. (Hexadecimal notation is merely a form of shorthand used to describe the binary, or base 2, condition of each bit in memory contents and addresses. It uses the single digits 0 through 9 and A through F to represent the numbers 0 through 15. Each digit represents 4 bits, two digits represent 8 bits or one byte, and four digits represent 16 bits or two bytes. For a complete explanation of hexadecimal notation, refer to the Editor/Assembler manual or another text concerning Assembly Language.) The third column, separated from the second by a colon (":"), shows the hexadecimal representation of the contents of the two bytes of memory found at each address. The fourth column contains the Assembly Language translation of the memory contents shown in the third column.

Let's give it a try. Enter 0 as the starting address and 50 as the ending address. As the program works, practice stopping (space bar) and restarting ("enter" key) the data. When the program stops of its own accord at 50, don't "PRESS ENTER TO CONTINUE" yet. Look at the line that starts with 44. That's the decimal address of a byte in low memory. The hexadecimal representation of that address is in the second column. (44 decimal=002C hexadecimal). With most 99/4A's, the third column will contain 020F, the hexadecimal representation of the contents of the two bytes of memory found at address 44 and address 45. Remember, we look at two bytes at a time, starting with even numbers. The final column will contain "LI", the Assembly Language translation of 020F. The "LI" stands for "Load Immediate", and instructs the computer to load into a memory register the value immediately following—in this case 8C02, the memory contents found at address 46. Lines that lack a fourth column, such as 46, describe memory addresses that contain values (data) rather than Assembly Language statements.

When you are ready to proceed, press "enter" to continue, as the program requests. The screen will clear and you will be given the opportunity to enter new starting and ending addresses. Now you're ready to load your own Assembly Language programs into RAM and start disassembling them.

We have a few final suggestions. This program was intended to let you explore the limits of your computer—it will not prevent you from entering illegal values. If the program crashes while attempting to read an illegal address (not a value between -32768 and 32767) or trying to open a printer illegally, merely enter RUN and try again. There are also some locations (fortunately only a few) that are legal addresses but that may cause a program crash or even a system crash if you try to disassemble them. Texas Instruments apparently does not want you to look everywhere. We cannot tell you exactly where you should not look, because the forbidden addresses vary from machine to machine. However, even a system crash is not fatal. If the computer locks up, shut off your equipment and then turn it back on. (Do not leave a disk in the disk drive during this process.) Then reload the program, making a note of the address that caused the crash so that you can avoid it in the future.

Finally, please realize that a disassembler is a very obedient and uncritical servant. It will cheerfully try to disassemble anything it finds, starting wherever you tell it to, assuming that you have sent it to an Assembly Language program. When it finds 83E0 as the first 2 bytes in the area of memory you have told it to disassemble, it will tell you that that location contains the Assembly Language statement "C" ("Compare Words"). If you have sent it to a value instead of a statement, or have sent it into a BASIC program, or have started it on an odd address, the disassembler will be incorrect—it depends upon your direction to start it off in the right place.

If you follow these suggestions carefully, you will find the disassembler to be a valuable tool for learning more about your machine—and the phrase "strolling down memory lane" will take on an entirely new meaning for you.

A FEW NOTES CONCERNING PLOTTING

The Texas Instruments 99/4 and 99/4A Home Computers have high-resolution screens which resolve 192 pixels (pixel=picture cell—the smallest definable point on the screen)

vertically by 256 pixels horizontally. Unfortunately, neither console BASIC nor Extended BASIC includes a pixel plotting command. Instead, both BASIC's base their graphics on character definitions—characters (letters, numbers, etc.) can be redefined and then called to any location on the 24 × 32 position character grid. As characters are redefined for graphics, no reference is made to the 192 × 256 pixel grid. Consequently, plotting individual pixels or points at a precise location on the screen is a laborious process. Extended BASIC provides some relief from this dilemma—points can be plotted on the 192 × 256 grid using sprites. Unfortunately, no more than 28 sprites can be used at a time. 28 points is sufficient for a few plotting purposes, but is not nearly enough for drawing.

Each of the three plotting programs or subprograms included in UTILITIES I permits the plotting of up to several thousand points. This is a very powerful capability, but it is important to understand how it is achieved in order to make the best use of the program. Each UTILITIES I plotter works by redefining characters. Depending on the program, there are 110 to 126 redefinable characters. Each time a point is plotted in a previously blank character position, one of those redefinable characters is used. When a point is plotted in a character position that has already been defined, that character is **redefined** and a new character is **not** used. Consequently, if you plot points at widely scattered locations on the screen, it is possible to run out of characters after plotting as few as 110 points. On the other hand, if you plot points in a very limited area, it is possible to plot as many as 8064 separate points. Typical applications usually permit 500 to 1,000 points to be plotted before all redefinable characters are used.

When no additional redefinable characters are available, each plotter continues to allow points to be called (requested by the primary program). If a called point is in an undefined character position, the plotter cannot plot the point on the screen. Instead, it sounds a tone and returns control to the calling program or routine. If the point called is in a previously redefined character position, the point is plotted on the screen before control is returned to the calling program or routine.

Because these plotters work by redefining characters, they may interfere with regular characters that you have displayed on the screen. If you plot a point inside the 8 × 8 pixel matrix of a displayed character, the point will appear not only where you

have requested it, but also in the same relative position inside every repetition of that character on the screen. (If you plot a point in the center of a "C", every "C" on the screen will show that point.) Additionally, as the plotters use up redefinable characters, they may take over and redefine characters you have displayed on the screen. If this happens, your displayed characters will begin disappearing or looking very strange. The console and Extended BASIC versions start by defining undefined characters, so they won't interfere with regular characters until the supply of undefined characters is exhausted. The Assembly Language version, for technical reasons, starts using regular display characters immediately.

The TI computers measure X (vertical) values from the top of the screen and Y (horizontal) values from the left of the screen. We recommend that you use integer values when calling points to the plot routines; you may experience difficulty if you fail to do this. And if you use any of the plotters for line drawing, remember that diagonal lines use up redefinable characters (and, therefore, points) faster than vertical or horizontal lines.

Note that the screen resolution for monitors or television sets is 192 × 256 pixels no matter how wide or tall the screen. All screens will display more points horizontally than vertically; however, most screens will display more points **per inch** vertically than they do horizontally. As a result a circle, square, or other uniform figure drawn on the screen will appear to be wider than it is tall. The effect will vary from screen to screen, but if uniformity in the vertical and horizontal dimensions of a displayed figure is important, you may wish to compensate for this distortion by making the figure taller (in number of plotted points) than it is wide.

A FEW NOTES CONCERNING SCREEN DUMPS

All printers are capable of printing standard characters (letters, numbers, etc.) as they are sent to it by your computer. You may have noticed however, that the characters printed by the printer do not necessarily look exactly like those displayed on the computer's screen. A lower case "a", for example, may look like

a small capital "A" on the screen, yet be printed looking like the more traditional lower case "a's" on this page. The reason for this difference of appearance is that the computer is not sending the printer a graphic description of the character to be printed; instead, it is sending the standard character number (96=lower case "a") of the character to be printed. The printer looks up this number in its own internal tables, finds a graphic description of a character that corresponds to that number, and prints that character.

You can prove this to yourself by entering the following program:

```
100  CALL HCHAR(1,1,65,768)
110  OPEN #1:"RS232.BA=4800.
     PA=N.DA=8"                  —or whatever your printer
                                  characteristics are
120  FOR I=1 TO 24
130  FOR J=1 TO 32
140  CALL GCHAR (I,J,A)
150  PRINT#1:CHR$(A);
160  NEXT J
170  PRINT #1:""
180  NEXT I
190  CLOSE #1
```

If you RUN this pseudo-screen dump program, it will fill the screen with capital "A's", and then send the screen contents to the printer. Try it. Looks pretty good, doesn't it? Unfortunately, it will work only as long as standard character definitions are used. If you enter the following line, which redefines character 65 (usually capital "A") to look like a diagonal line, you will be able to see the problem:

```
90  CALL CHAR(65,"8040201008040201")
```

When you RUN the program with this additional line, the screen displays our new character definition, but the printer only knows that the computer is repeatedly sending character 65, and prints another batch of capital "A's".

Use of this common method allows great standardization among computers and printers. It also permits faster and more economical exchange of information—the standard character number itself contains only about one-eighth as many bits of data as a complete graphic description of a character. However, as we have seen, this standard method does not allow the transmission of graphic data.

In order to take advantage of the superior graphics abilities of state-of-the-art microcomputers such as the Texas Instruments 99/4 and 99/4A, many printer manufacturers are now including special (and frequently dissimilar) graphics modes in some of their products. All of the screen dumps included in UTILITIES I take advantage of the special graphics mode shared by the TI 80-column printer and Epson printers with Graftrax or Graftrax Plus. They may also work with other printers, but these instructions are limited to the requirements of the Texas Instruments and Epson printers.

Your printer has a number of internal switches on its serial interface (RS232) circuit board that control the baud (data transmission) rate, number of data bits sent with each transmission, graphics capabilities, etc. The switch positions usually set at the factory are not compatible with graphics; in order to take advantage of the screen dump utilities in this package, you will probably need to change some of the settings.

Consult your printer manual to find the appropriate procedure for gaining access to these internal switches. Be sure to unplug the printer before opening the case, and follow any other safety precautions recommended by the manufacturer. Read the switch settings. Slide switches are easy to read, but most printers have small push-type switches. Remember that the little red markings present on some of these push-type switches are not switch-position indicators, but are intended to show the part of the switch that must be pushed to change its setting; if read as switch position indicators, they show the opposite of the true settings. The side of the switch that is depressed (the down side) indicates the side to which the switch is set.

It is possible to use the screen dumps without following all of the recommendations below, but some are absolutely necessary, and others will allow all of these instructions to apply uniformly to TI and Epson printers. **If you know what you are doing,** you may wish to modify your own procedures when you have finished the practice exercises in this manual.

The Texas Instruments 80-column printer requires the following settings:

SW 1-1	OFF	SW 2-1	OFF
SW 1-2	ON	SW 2-2	ON
SW 1-3	OFF	SW 2-3	OFF
SW 1-4	OFF	SW 2-4	ON
SW 1-5	ON		
SW 1-6	ON		
SW 1-7	ON		
SW 1-8	OFF		

Most Epson printers require changes only in the single bank of 8 switches:

S1 ON
S2 OFF
S3 OFF
S4 OFF
S5 ON
S6 ON
S7 OFF
S8 ON

These positions prepare your printer to send 8 bits of data for each character at a rate of 4800 baud (bits per second), and instruct it not to check for parity errors (essential for graphics). This will let your printer and computer exchange graphic data and do so at a faster rate than the factory settings allow.

Your printer may require different switch settings than those shown here in order to allow it to receive 8 bits of data per character at 4800 baud and to disable parity checking. If so, consult your printer's manual for further guidance. Those who have connected their printers via the parallel port will also need to deviate from these instructions. Parallel ports don't allow baud rate changes (they always go at top speed); they almost always send 8 data bits; and they don't usually permit parity checking. Consult your printer's manual to be sure, but you may not need to make any switch setting changes. However, you will need to make some software changes. Substitute the appropriate parallel

port description for RS232.BA=4800.PA=N.DA=8 in the example programs in this book and in the console BASIC (line 1050) and Extended BASIC (line 10020) screen dump routines.

Setting these switch-positions will make some changes in your life. Your printer's description is now RS232.BA=4800.PA=N.DA=8. You must use this description instead of the shorter RS232 whenever you OPEN your printer in a program or LIST programs to it. That's really only a slight penalty to pay for the increased capability and higher speed you now have available.

Please read the portion of the preceeding section (A Few Notes Concerning Plotting) that pertains to the variations in vertical and horizontal density of points. (It's in the last paragraph of that section.) The same variations that affect screen displays may be even more pronounced on a printer, and may necessitate additional corrections in your plotting efforts if you wish to simulate symmetry in vertical and horizontal dimensions.

CONSOLE BASIC PLOT AND SCREEN DUMP (PLOTDUMP)

Required Equipment

- none

Optional Equipment

- RS232 Interface with compatible printer (required for screen dump)
- Joysticks

Instructions

Select console BASIC. Load the program normally. Enter RUN.

After the logo screen appears and clears, you are asked whether you wish to use the joysticks or the keyboard to control plotting. (The joysticks are a little faster, but the keyboard gives more precise control.) After you enter your choice, the program enters plotting mode. A one-pixel cursor now exists at position 96,128 (the center of the screen), although you cannot see it yet.

Three keys control the cursor's condition. The "7" key turns the cursor on. When the cursor is on, it plots a point wherever it is located. If you move the cursor while it is on, it plots points continuously, leaving a line in its wake. The "6" key turns the cursor off. The "8" key puts the cursor into erase mode. In erase mode, the cursor plots a one-pixel space wherever it is located. If there is a point at that location, the point is erased. Don't touch the "9" key. We will discuss its use a little later.

Go ahead and press the "7" key to make the cursor appear. The program sounds a note to let you know that it has accepted input from a key. Move the cursor with joystick #1 or the keyboard controls. Operation of the joystick controls is intuitive: up, down, right, left, and diagonal positions move the cursor in those directions. (If you are using a 99/4A, make sure the "alpha lock" key is in the up, or off, position.) The keyboard controls are equally obvious. The "S", "D", "E", and "X" keys control, respectively, left, right, up, and down movement of the cursor. The intermediate keys "W", "R", "C", and "Z" control diagonal movement.

The program sounds a note every time the cursor is told, by the keyboard or joystick, to move one pixel position, so you can trace its approximate movement even with the cursor turned off. To locate it exactly when it is turned off, press "7" to make it appear. If it is not precisely where you want it, press "8" to erase the unwanted point, and then "6" to turn the cursor off. (It is better to leave the cursor off than in the erase mode if you wish to move it without plotting—it will move faster and use up no redefinable characters.) Then you can move the cursor to the correct location, counting pixel positions, and repeating the "7", "8", "6" (on, erase, off) procedure if necessary.

As you are plotting, cursor movement may stop for a few seconds. This doesn't mean your computer has "locked-up" or "crashed". Instead, the computer is stopping momentarily for what is termed "garbage collection", or reallotment of memory space. This program uses a large amount of memory space for variables and then moves these variables around in memory as characters are redefined; because so much data is being moved from one place to another in memory, more frequent "garbage collection" is necessary.

Pressing the "9" key tells the program to execute a screen dump. If you do not have an RS232 Interface and printer, pressing "9" will crash the program when the computer tries to

"open" a non-existent printer. To avoid this problem, those without printers should insert REM at the beginning of line 1840, the line that "calls" the screen dump subroutine, and SAVE the program with this change. If you acquire a printer at a later date, deleting the REM will reactivate that line.

For those with RS232 Interface and Compatible Printer

Once you have put something on the screen that you admire, it's time to try the screen dump. Make sure the printer is turned on and then press the "9" key. When the screen dump is completed, you are returned to your plotting right where you left off; if you wish, you can embellish the screen image and then dump it again.

Why is the image printed sideways? Good question. Epson and Texas Instruments printers define graphic characters as they print: from left to right. The TI computers, on the other hand, define graphic characters from the top to the bottom. Printing the screen sideways allows the computer graphics to conform more closely to the printer graphics and reduces the amount of data manipulation necessary. As a consequence, printing the screen sideways takes only about one-third of the time it would take to print it "right side up".

Additional Information for Experienced Programmers Only

Console BASIC does not permit merging of subprograms or subroutines into other programs. Additionally, console BASIC does not support the CALL CHARPAT command which allows Extended BASIC programs to directly read the character definition tables in memory. These limitations necessitate the complicated and memory-intensive practice of maintaining all graphic character definitions in memory in two separate locations: in the character definition tables where the screen display can use them and in a string array (A$()) where the console BASIC program can manipulate them. For these reasons, this utility has been designed as a "stand-alone" program. However, its primary plot and screendump routines can be incorporated into your personal programs even though they were not intended for this purpose.

The plotting routine extends from line number 130 to 1040. In addition to these lines, line 110 must be included in your program as a lower line number than the first line of the plot subroutine, and line numbers 1530 to 1550 must be included in your program and executed once before the first call of the plot routine. To plot a point: have your program define the variable "X" as an integer between 1 and 192 and the variable "Y" as an integer between 1 and 256; then set E$="1" (or to "0" if you wish to erase a pixel). When you GOSUB to the plot routine, it will plot a point at the X,Y location your program specified.

The screendump routine extends from line number 1050 to line number 1370. A GOSUB to the first line of this routine will execute a screen dump. Remember that the screen dump gets its graphic information from the A$() array, not from the regular character definition tables, which are unavailable in console BASIC. For a screen to be dumped to a printer without errors, every character on the screen must be a space (standard character number 32) or a character (with a standard number between 34 and 143, inclusive) that has been defined in the A$() array.

Please ensure that your main program does not redefine any of the variables used by the routines—except, of course, A$(), X, Y, and E$.

EXTENDED BASIC PLOT AND SCREEN DUMP (PLOT and SCREENDUMP)

Required Equipment

- Extended BASIC

Optional Equipment

- RS232 Interface and compatible printer (required for screen dump)

Instructions

Make sure that the Extended BASIC cartridge is properly inserted into the console. Select Extended BASIC.

Both of these routines are intended for use as subprograms that can be incorporated into and called by your own primary programs. Extended BASIC does not support the MERGE capability for programs stored on cassette. For this reason, the cassette-based versions of these routines have been combined and saved as a single Extended BASIC program. Load this program normally before entering your own program from the keyboard. The disk-based versions of these routines are saved separately in MERGE format. Before or after you enter your own program, load these programs by entering MERGE DSK1.PLOT and/or MERGE DSK1.SCREENDUMP. Line numbers for PLOT and SCREENDUMP start at 10000, so you should use lower line numbers for your primary programs to avoid unintended line deletions when these routines are MERGEd.

Enter NEW and then load the routine(s) as described above. Enter the following lines:

```
100 CALL CLEAR::FOR I=1 TO 192 ::CALL
    PLOT(I,I,"1")::NEXT I
110 GOTO 110
```

RUN the program. Line 100 clears the screen, then draws a diagonal line; line 110 holds it on the screen. You can now call PLOT (or SCREENDUMP—see below) just as you would any other Extended BASIC subprogram such as HCHAR or CLEAR. It requires that three variables, separated by commas, be passed to it in parentheses. The first two variables are the X (integers from 1 to 192) and Y (integers from 1 to 256) coordinates of the point you wish to affect. The third variable is a string variable that determines whether the pixel at that location will be plotted ("1") or erased ("0").

Break ("shift C" or "function 4") the current program and enter the following program. (Just type over the previous lines 100 and 110. If you enter NEW, you'll have to reload the utilities. Enter line 110 only if you have the RS232 interface and compatible printer.)

```
100 CALL CLEAR :: R=57.29578 :: FOR I=0 TO 2*PI
    STEP .0174524 :: X=INT(SIN(I)*R) :: Y=
    INT(COS(I)*R) :: CALL PLOT(92-X,128+Y,"1") ::
    NEXT I
110 CALL SCREENDUMP
120 GOTO 120
```

If you have a printer, make sure that it is turned on at this time. RUN the program. It will plot a circle on the screen and then, if you have the appropriate equipment, it will dump the screen image to your printer. This screen dump utility does not require that any variables be passed to it from your primary program.

That's about all that there is to it. The plot routine can be combined with joystick or keyboard commands and used for drawing (as in the console BASIC plot and screen dump program) or combined with labels and mathematical algorithms for scatterplots or other high-resolution plotting purposes. The screen dump routine is always ready to transfer the contents of any screen, no matter how created, directly to your printer. Other possibilities are limited only by your imagination.

ASSEMBLY LANGUAGE PLOT AND SCREEN DUMP (PDMLDISK)

Required Equipment

- Extended BASIC
- Expansion RAM

Optional Equipment

- RS232 Interface and compatible printer (required for screen dump)

Instructions

Make sure that the Extended BASIC cartridge is properly inserted into the console. Select Extended BASIC.

Both of these Assembly Language routines are combined into one Assembly Language program; they are intended for use as subroutines that can be loaded into memory and called by your own programs. The cassette-based version is recorded as a program that must be loaded and stored **before** you enter your own Extended BASIC or Assembly Language programs. Load it normally. Once it is loaded, enter RUN to store it correctly. After the cursor returns, enter NEW. To load and store the disk-based version, enter CALL INIT. Then enter CALL LOAD("DSK1.PDMLDISK").

Once the Assembly Language program is properly loaded and stored, entering CALL INIT will erase it from memory. Be careful.

Enter NEW and then load the Assembly Language program as described above. Enter the following lines:

```
100  CALL LINK("PLOT",1,1,2)
110  FOR I=0 TO 191 :: CALL LINK("PLOT",1,1,1) ::
     NEXT I
120  GOTO 120
```

RUN the program. Line 100 initializes the plotter and clears all character definitions; this statement must be executed to give the plot routine its full complement of points for plotting. Merely halting the BASIC program that calls it will not reset an Assembly Language routine. Line 110 draws a diagonal line on the screen; line 120 holds it there. You can now link to PLOT (and the screen dump, called PRINT—see below) very easily. PLOT requires that three numeric variables, separated by commas, be passed to it in parentheses. The first two variables are the Y (integers from 0 to 255) and X (integers from 0 to 191) intersection of the point you wish to affect. (Note the reversed order of X and Y). The third variable determines whether the pixel at that location will be plotted (1) or erased (0).

Break ("shift C" or "function 4") the current program and enter NEW. Enter the following program, omitting lines 120 and 130 if you do not have an RS232 Interface and compatible printer.

```
100  CALL LINK("PLOT",1,1,2)
110  R=57.29578 :: FOR I=0 TO 2*PI STEP .0174524 ::
     X=INT(SIN(I)*R) :: Y=INT(COS(I)*R) :: CALL
     LINK ("PLOT",128+Y,92-X,1) :: NEXT I
120  P$="RS232.BA=4800.PA=N.DA=8.CRLF"
130  CALL LINK("PRINT",P$)
140  GOTO 140
```

If you have a printer, make sure it is turned on. RUN the program. It will plot a circle on the screen and then, if you have the appropriate equipment, it will dump the screen to your printer. (This screen dump prints right side up!) The screen dump utility requires that the printer description be passed to it as a string. Note that the printer description has CRLF appended to

it. If you do not include this in your printer description, the computer will cause the printer to skip a number of extra lines and introduce unwanted data as it prints the screen contents.

The Assembly Language program also supports saving screen contents to disk and then reloading the data to the screen. If you have a disk system, break ("shift C" or "function 4") the program in memory and add the following lines:

```
132  CALL LINK("SAVE","DSK1.PICTURE")
135  CALL CLEAR
138  CALL LINK("OLD","DSK1.PICTURE")
```

RUN the program. It will now draw the circle, dump it to your printer (if you have one), and save the screen image to disk drive 1 as a file named PICTURE. (You can use any name you wish for the file name.) It will then clear the screen and recall the image from the file. This is a very useful capability.

Now you can write some programs of your own, and start LINKing.

SPEECH UTILITY (SPEAKOUT1)

Required Equipment

- Extended BASIC
- Speech Synthesizer

Optional Equipment

- Speech Editor
- RS232 Interface and printer

Preliminary Information

Spoken English words are constructed of standard units of speech called phonemes. Each phoneme represents a standard pronunciation of a letter (such as a consonant or a long or short vowel sound) or a common combination of letters (such as "ch" or "ng"). The phonemes used to represent a word are not dependent on the spelling of the word. The word "kite" has four letters, but only three phonemes ("k", long "i", and "t") are

necessary to describe its pronunciation. "Tough" has even more letters, but requires only the phonemes "t", short "u", and "f" to show its correct pronunciation. Dictionaries use letters and special symbols (such as an upside down "e") to represent the 45 standard phonemes in their pronunciation guides; to avoid the difficulties involved in entering special symbols from your keyboard, this program has assigned numbers to each of the phonemes:

1. a as in **fat**	24. f as in **fall**
2. a as in **date**	25. g as in **get**
3. a as in **bare**	26. h as in **he**
4. au as in **auto**	27. j as in **jump**
5. e as in **ten**	28. k as in **kill**
6. e as in **meet**	29. l as in **let**
7. e as in **here**	30. m as in **met**
8. e(r) as in **over**	31. n as in **not**
9. i as in **hit**	32. p as in **put**
10. i as in **bite**	33. r as in **red**
11. o as in **top**	34. s as in **sell**
12. o as in **go**	35. t as in **top**
13. o as in **fork**	36. v as in **vat**
14. oo as in **tool**	37. w as in **will**
15. oo as in **book**	38. y as in **yet**
16. oi as in **oil**	39. z as in **zebra**
17. ou as in **out**	40. ch as in **chin**
18. u as in **up**	41. ng as in **ring**
19. u as in **cute**	42. sh as in **she**
20. u as in **turn**	43. th as in **thin**
21. a as in **ago**	44. th as in **then**
22. b as in **bed**	45. zh as in **azure**
23. d as in **did**	

This speech utility allows you to enter the phoneme numbers that describe a word, listen to the word, improve its pronunciation by the computer, and then print (on a printer or on the screen) the speech data that tells your computer how to

pronounce the word. This speech data can then be incorporated into your own Extended BASIC programs (or console BASIC programs if you have the Speech Editor cartridge) for later pronunciation by your computer.

Instructions

Make sure the Extended BASIC cartridge is properly inserted into the console. Select Extended BASIC. Load the program normally. Enter RUN. After the logo screen appears and clears, you are asked to enter phoneme numbers. Referring to the table above, enter these numbers, following each with a carriage return ("enter" key) : 28, 29, 9, 41, 11, 31, and 39. Now enter a 99 (or any number other than a correct phoneme number) to signal the program that you are finished entering phoneme numbers. The program generates the speech data for the word you have created, says the word aloud, and displays the main menu.

The first choice on that menu tells the program to speak the word again. Press the "1" key, then the "enter" key. The program pronounces the word again and returns to the main menu. Note, however, that your previous selection (1) is now displayed as the default value for "Your Choice". All entered choices are retained by this program as default values. This enables you to repeat actions merely by pressing the "enter" key, and prevents you from having to re-key lengthy values such as printer description (see below).

Press "enter" several times. The word sounds recognizably like "Klingons", the bad guys on the old science fiction television show. You can make the word sound a little better, though. When the main menu is displayed again, enter "2", to select "SMOOTH WORD". Smoothing a word consists of removing speech data (measured in bytes) from between the component phonemes to make the word "flow" a little better. The number of bytes to remove here is purely a matter of taste. Up to a certain point, which varies from word to word, the pronunciation sounds better as additional bytes are removed; beyond that point, it begins to sound worse until it becomes unrecognizable. Try several values, but 5 may be your best choice for this word.

You can skip over choice number 3 on the main menu for now—you've already seen what it is like to enter phonemes. Try number 4, and when its sub-menu is displayed, select 1, the screen, as the place to list the phoneme numbers. You don't need

to know the phoneme numbers used for the word "Klingons"—they're printed right in this manual. But if you enter a new word and like the way it sounds, this listing quickly summarizes the input data that created the word. It's especially handy as a heading for speech data listed on a printer. (See below). Press "enter" to return to the main menu.

Enter number 5. You have the same listing choices for speech data that you did for the phoneme numbers. Press "enter" to choose the screen. To halt the listing press the space bar; to continue it, press "enter". That's a lot of data! When the listing is completed, press "enter" to return to the main menu. If you don't have a printer, skip the next paragraph.

Press "enter" again to return to speech data listing options. Enter 3 as your choice, and then enter your printer characteristics—an appropriate example is given on the line above the cursor. (If you press "enter" without typing printer characteristics, you are returned directly to the primary menu.) When the listing is completed, you are automatically returned to the primary menu.

If you have a disk drive, you can also save speech data on a disk. Press "enter" one more time to return to the speech data listing options, and enter 2 as your choice. Enter DSK1.FILE (or another valid file name) as your disk name. (If you press "enter" without typing a file name, you are returned directly to the primary menu.) When file storage is completed, you return to the main menu automatically.

The only choice left is number 6. Select 6. The screen you now see is part of a fail-safe routine. Unless you change the default "N" to a "Y", you are returned to the main menu when you press "enter". Press the "Y" key and then press "enter".

There are two ways to use speech data in a program. If you have saved the data on a disk, it can be recalled by a program and spoken. Here is a short example (Don't forget to enter NEW first):

```
100  B$="" :: OPEN #1:"DSK1.FILE", INTERNAL,
     VARIABLE 254 :: FOR I=1 TO 255 :: INPUT
     #1:A$ :: B$=B$&A$ :: IF EOF(1) THEN 120
110  NEXT I
120  CLOSE #1 :: CALL SAY(,B$)
```

Whether you have a disk drive or not, the following method is better for permanently including speech data as a part of your

programs. To keep this example short, we'll use the (unsmoothed) data from only a single phoneme —number 28, the initial "k" sound from the demonstration word. Enter NEW and then enter the following lines:

```
100 FOR I=1 TO 17 :: READ A :: B$=B$&CHR$(A) ::
    NEXT I
110 CALL SAY(,B$)
120 DATA 96,0,14,0,144,128,226,217,18,144,189,137,
    3,114,80,126,234
```

Line 100 initializes B$ and then reads the speech data one data element at a time—17 is the number of data elements in this example. Each time it reads a data element, it adds its string value to B$ with the CHR$ command. When it is finished reading, B$ contains the speech data necessary to say the "k" sound.

There is a limit placed by the computer on the amount of speech data that can be included in one word—255 bytes. If the phonemes you enter generate too much speech data, the program prints the message "WORD IS TOO LONG", shortens the speech data and the list of phoneme numbers to the allowable limit, and then proceeds normally.

The addition of new words to your speech synthesizer's vocabulary is a powerful capability. This program has been carefully written to make the very complex speech creation process as easy as possible. We hope this new power of speech is both useful and enjoyable to you.

A Final Note From REPUBLIC SOFTWARE

We thank you for your decision to purchase UTILITIES I. We have worked hard to make this software package useful to you as well as easy to use. However, if you have any questions or suggestions concerning the software or these instructions, please feel free to contact us.

REPUBLIC SOFTWARE, INC.
P.O. Box 23042
L'Enfant Plaza
Washington, D.C. 20024